PENCIL II

COMPUTER CONSOLE

USER'S MANUAL

PART I OPERATION MANUAL
PART II SD-BASIC REFERENCE MANUAL

PART I

OPERATION MANUAL

Specifications and information of this manual are subject to change without notice.

All efforts have been made to supply accurate and complete information in this manual. Soundic Electronics Limited assumes no responsibility for any errors in this manual or their consequences.

PENCIL is a trademark of Soundic Electronics Limited

First Edition 1983
Printed in Hong Kong

The equipment described in this Operation Manual generates and uses radio frequency energy. If not installed and used properly in strict accordance with our instructions, it may cause interference to radio and television reception.

In case of suspected interference caused by the computer, switch off the computer, if the interference discontinues, it was probably caused by the computer. To rectify the interference, user is recommended on all or any of the following measures:

- Relocate the direction of the TV or radio antenna until the interference stops.
- Set the computer to one side or the other of the TV set or radio.
- Move the computer away from the TV set or radio.
- Connect the computer to a different power outlet so that the computer and receivers are on different branch circuits.
- If all fail, consult the dealer or an experienced radio/ television technician for additional recommendations.

CONTENTS

1 INTRODUCTION

Dear Users,

Congratulation! You have just made the right choice in selecting this remarkable product.

Your PENCIL II is a sophisticated, powerful and easy-to--operate personal computer. It is designed with state-of-the-art microprocessor technology and manufactured under most stringent standard. Numerous versatile and unique features can be found which enable your PENCIL II to perform a wide range of exciting functions. It assists and enriches every aspect of your living, from business to education and home entertainments.

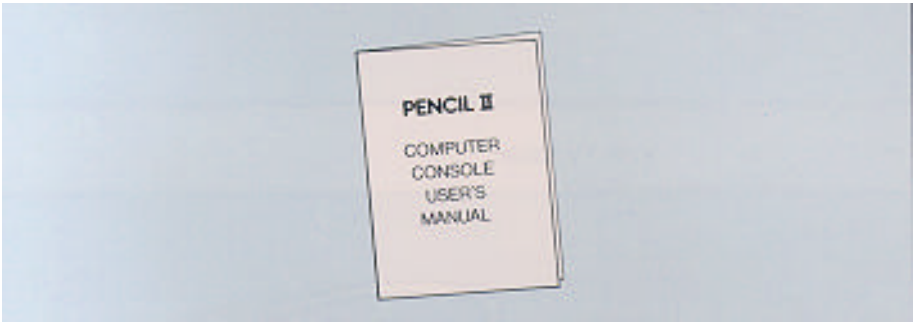
The Operation Manual is intended to explain the procedures in setting up and running of the PENCIL II. However, it does not teach programming. If you are a novice and/or not familiar with computer languages or programming, the SD-BASIC Reference Manual and other books about BASIC will be helpful to you.

For a good start, please read over this manual carefully and keep it safely in your possession for future reference.

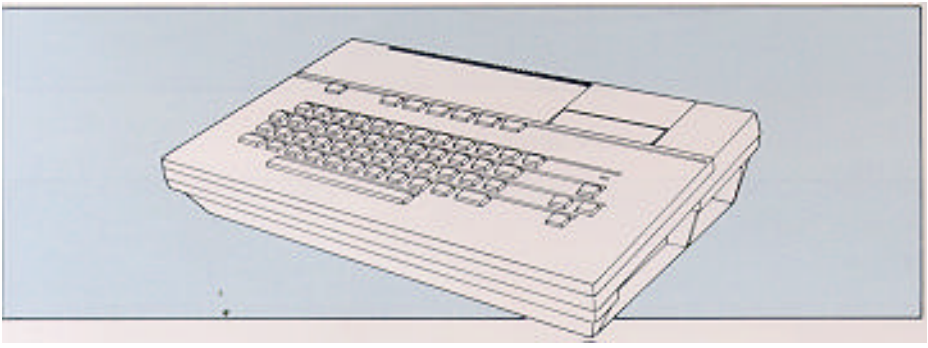
2 THE PENCIL II AND ITS ACCESSORIES

When you unpack the PENCIL 11, you should find the following enclosed:

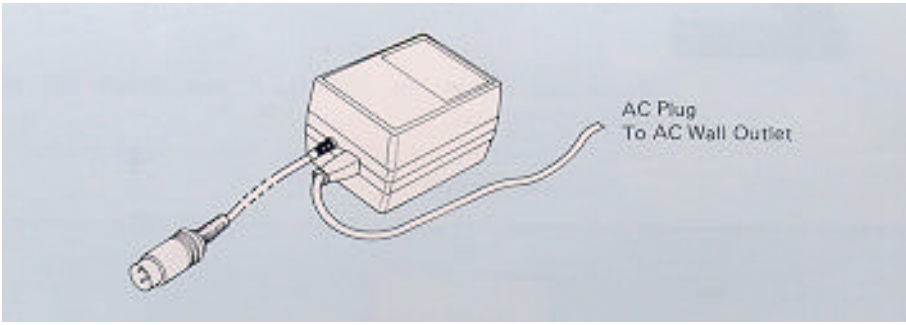
- An User's Manual (Operation Manual and SD-BASIC Reference Manual)



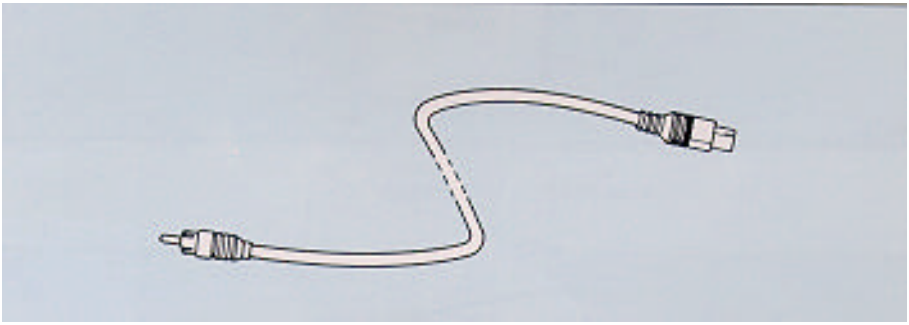
- A PENCIL II Computer



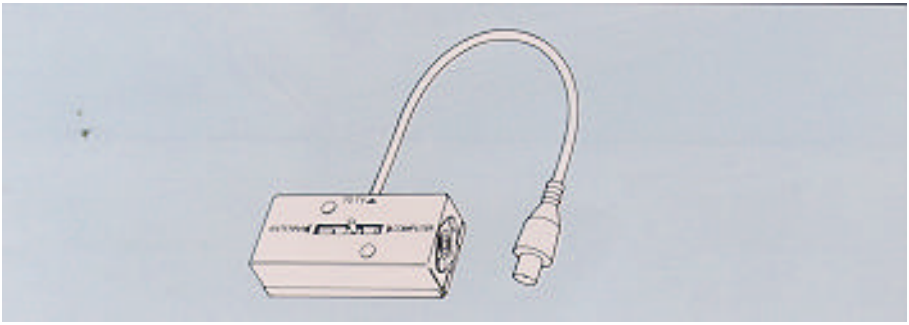
- A Power Supply Unit (PEN-902)



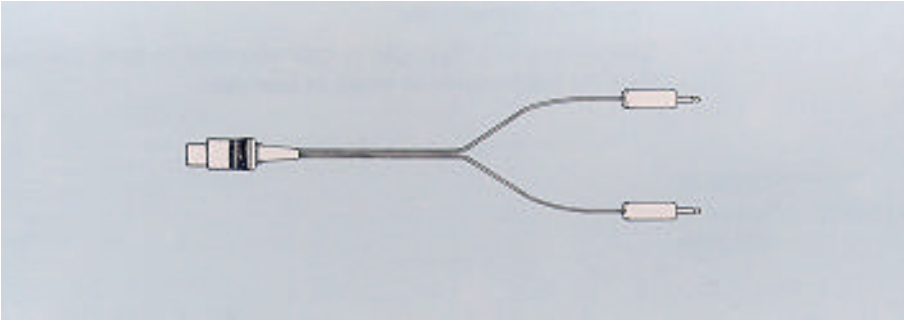
- A TV Cable



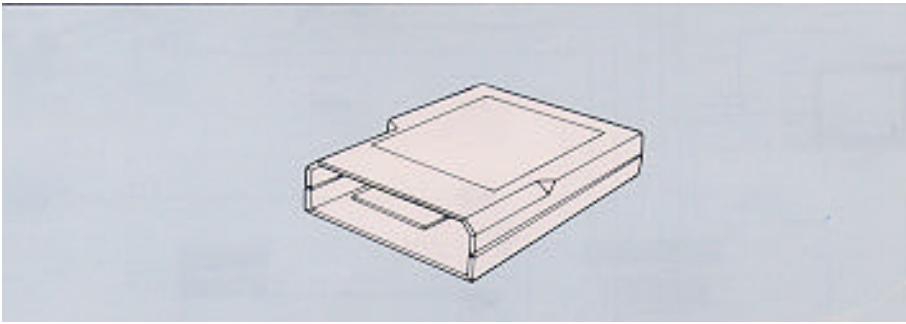
- A TV Switch Box (Optional)



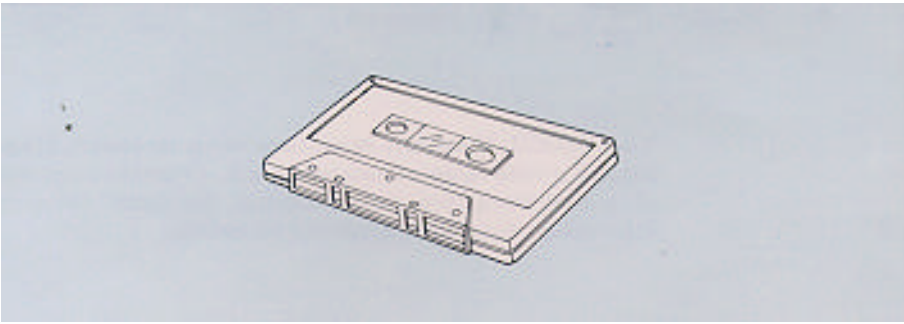
- A Cassette interface Cable



- A SD-BASIC Interpreter Cartridge



- A Demonstration Program Cassette

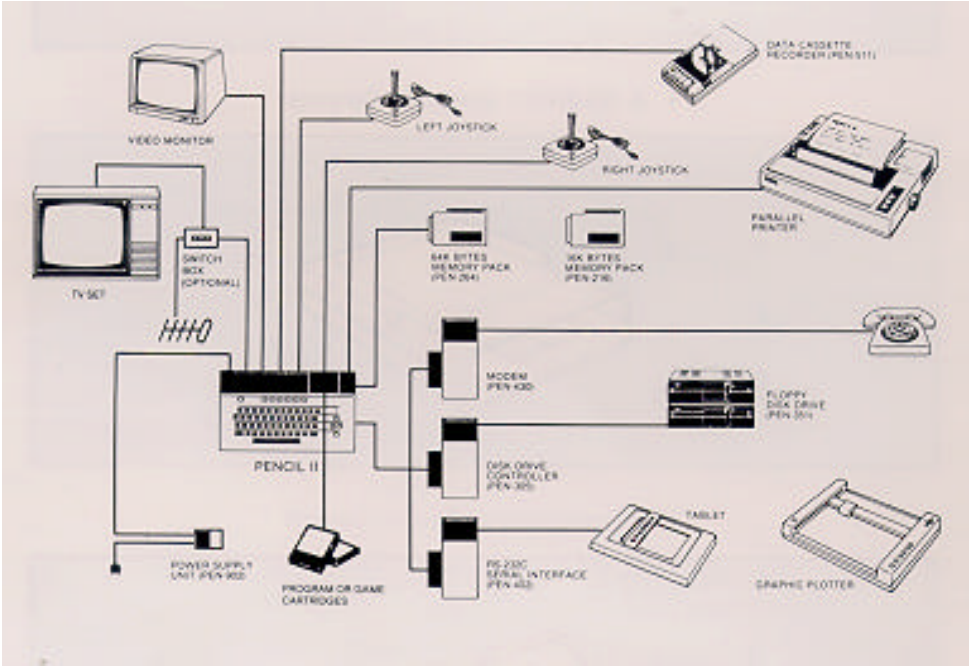


NOTE

Check all items. If any item is missing or damaged, contact your dealer immediately.

Save all packing materials in case you need to send this unit back for maintenance or repair in later days.

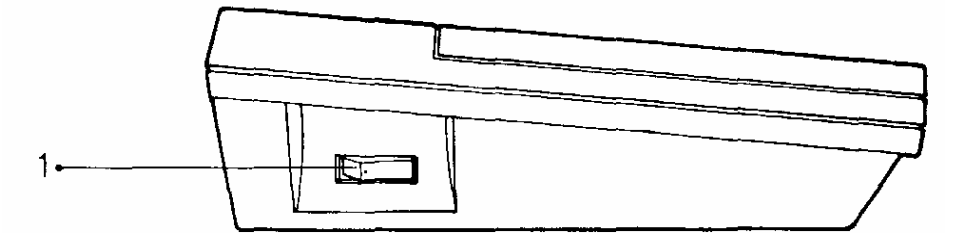
THE PENCIL II SYSTEM – Peripheral Diagram

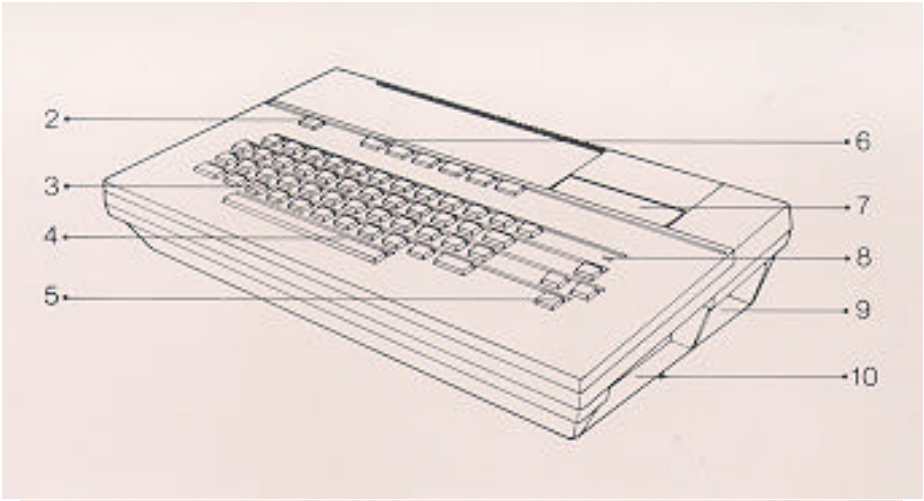


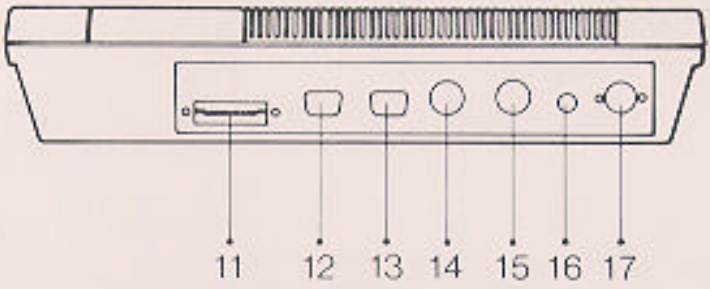
Your PENCIL II is a fully expandable computer system. It can support a wide range of peripherals and will satisfy your need of greater computing power. Consult the dealer for more information about these additional peripherals.

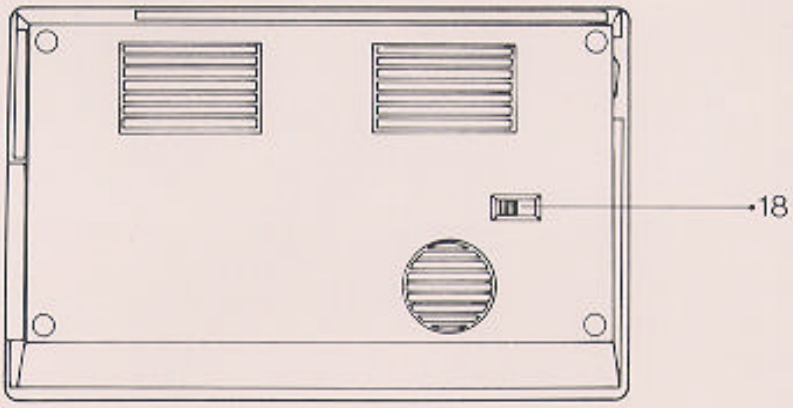
3 NOMENCLATURE

1. Power ON/OFF Switch
2. Reset Key
3. Alphanumeric Keyboard
4. Space Bar
5. Cursor Control Keys
6. User-defined Function Keys
7. Cartridge Door
8. Power Indicator
9. Memory Expansion Slot (for 16K and 64K Memory Pack)
10. Expansion Slot (with protective door)
11. Parallel Printer Port
12. Joystick Socket (right)
13. Joystick Socket (left)
14. Cassette Port
15. Video Monitor Socket
16. TV Socket
17. Power Socket
18. Channel Switch (optional)




2
3
4
5
6
7
8
9
10


11
12
13
14
15
16
17


18

4 HOW TO SET UP

In addition to the PENCIL II Computer Console and the supplied accessories, you will need:

- A regular TV set or composite video monitor for video display. (Both colour and monochrome can be used. However, no colour will be produced when using monochrome TV set or monitor.)

- A Data Cassette Recorder (PEN-511, optional) or any regular cassette tape recorder for loading and saving programs.

- For connection to other peripherals and accessories, such as game cartridges, disk drive units, parallel printers and telephone modems, etc, please refer to the separate instruction manuals supplied with these accessories.

Conecting of The Power Supply

The Power supply Unit (PEN-902) converts your AC electricity into the form that PENCIL II requires. Check the voltage of your house current and make sure it corresponds to the requirements stipulated on the label of the Power Supply Unit. Follow procedures listed below for proper connection:

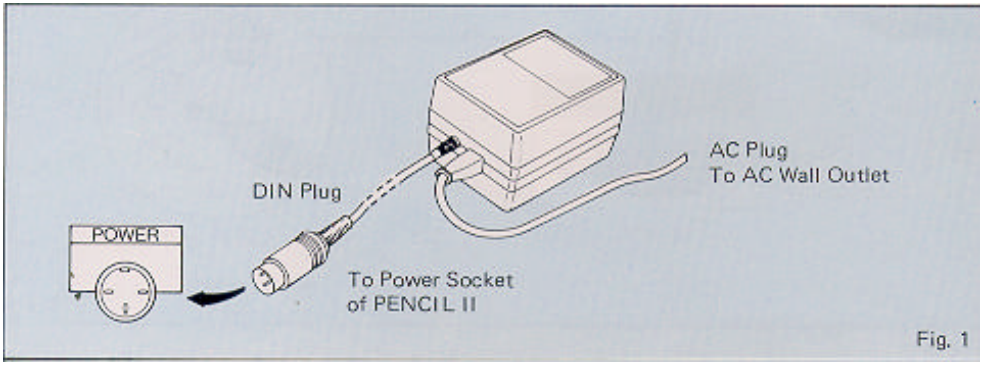


Fig. 1

Make sure the POWER switch is at the "OFF" position when connecting or disconnecting.

1. Insert the DIN plug of the Power Supply Unit into the Power Socket at the back of the PENCI L I I and the AC plug into the wall outiet of your house as shown in fig. 1.

CAUTION

Using a different power supply unit other than PEN-902 may cause severe damage to the PENCIL II. The warranty does not cover any damage incurred from such misuse.

Conecting to a Regular TV Set

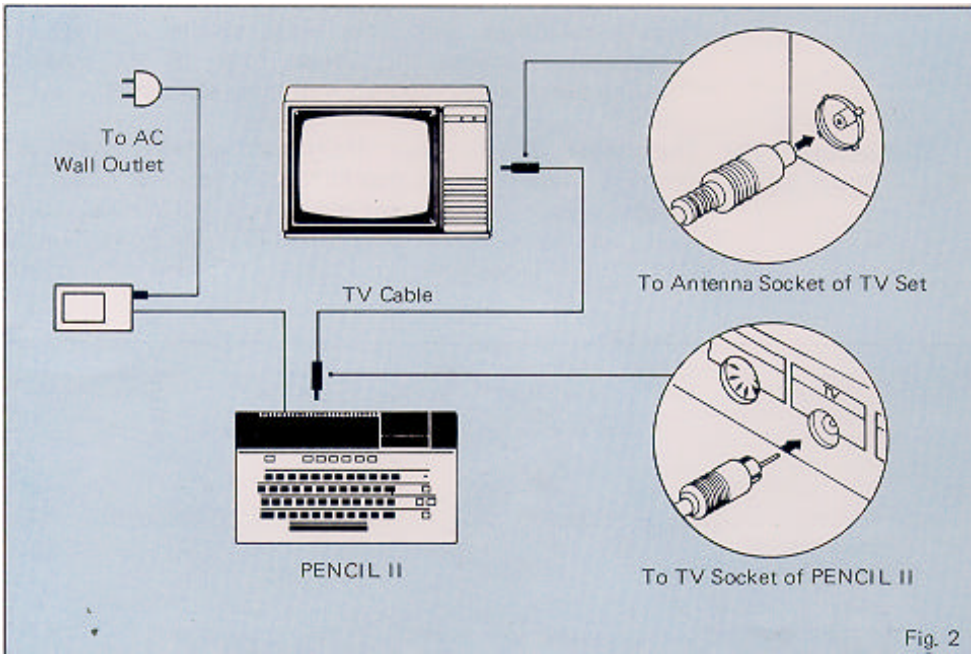


Fig. 2

1. Switch off both PENCI L II and TV set.

2. Connect the TV Cable to the TV Socket at the back of your PENCIL II and to the antenna socket at the back of the TV set as shown in fig. 2.

3. An optional Switch Box may be supplied with your PENCIL II package. The use of this Switch Box is to provide a convenient way of using your TV set for either computer use or receiving normal TV programmes by selecting with a switch. If you are going to use this Switch Box, make connections as shown in fig. 3.

NOTE
Please contact your dealer for the supply of this Switch Box.

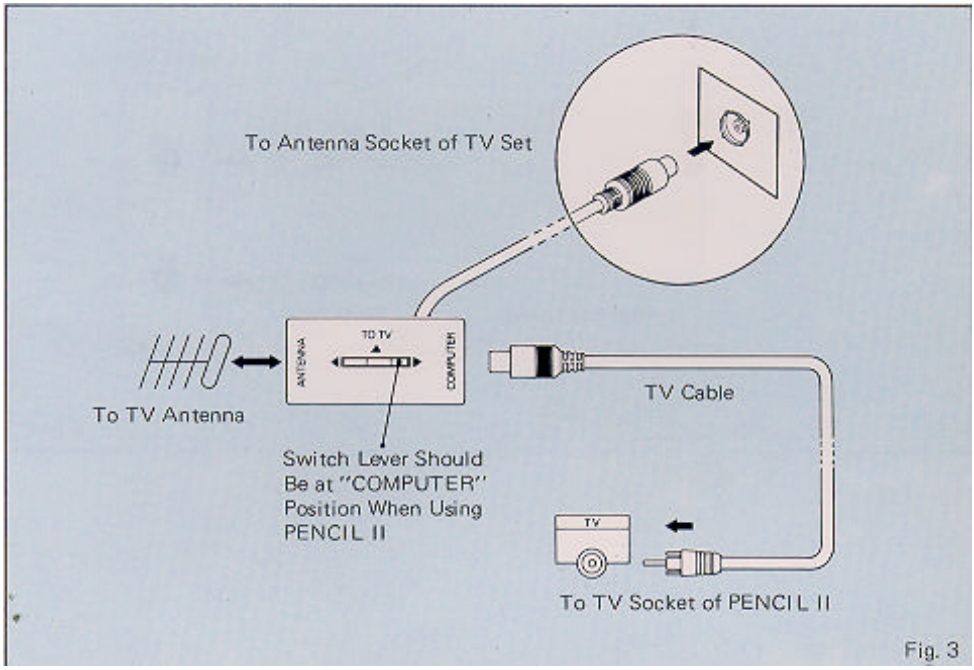


Fig. 3

Conecting to a Composite Video Monitor

1. For users using a composite video monitor with built-in audio system, procedures are similar to "CONNECTING TO REGULAR TV SET" except the following differences:

   a) Use the Monitor Cable (PEN-962), supplied separately, instead of the TV Cable.

   b) Insert the DIN plug of the cable into the Video Monitor Socket at the back of the PENCIL II, and the double RCA-plug into the Video and Audio Sockets at the back of your monitor as shown in fig. 4

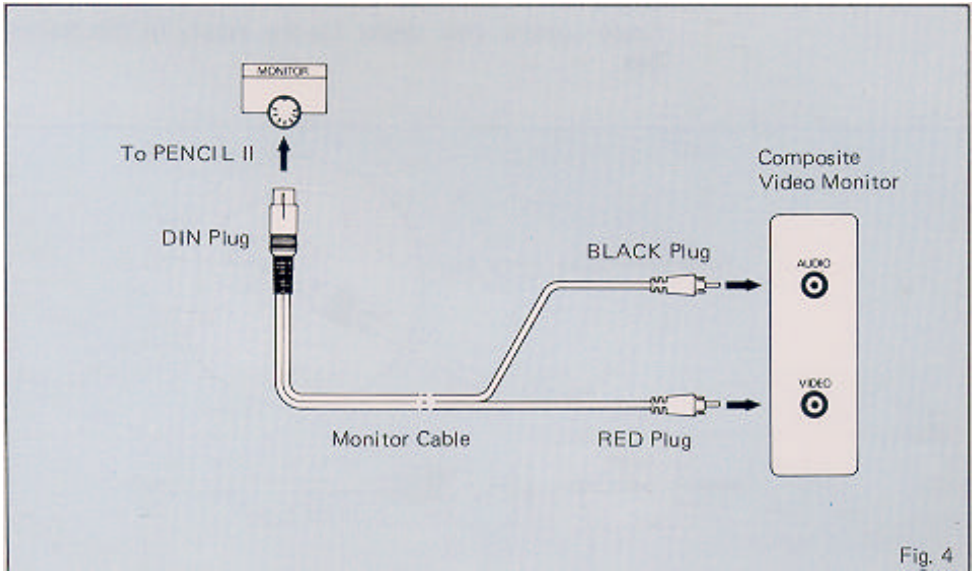


Fig. 4

2. If you are using a composite video monitor without built-in audio system, insert the RED plug of the Monitor Cable into the Video Socket at the back of the monitor as shown in fig. 5.

NOTE
In this case, the BLACK plug is left unused.

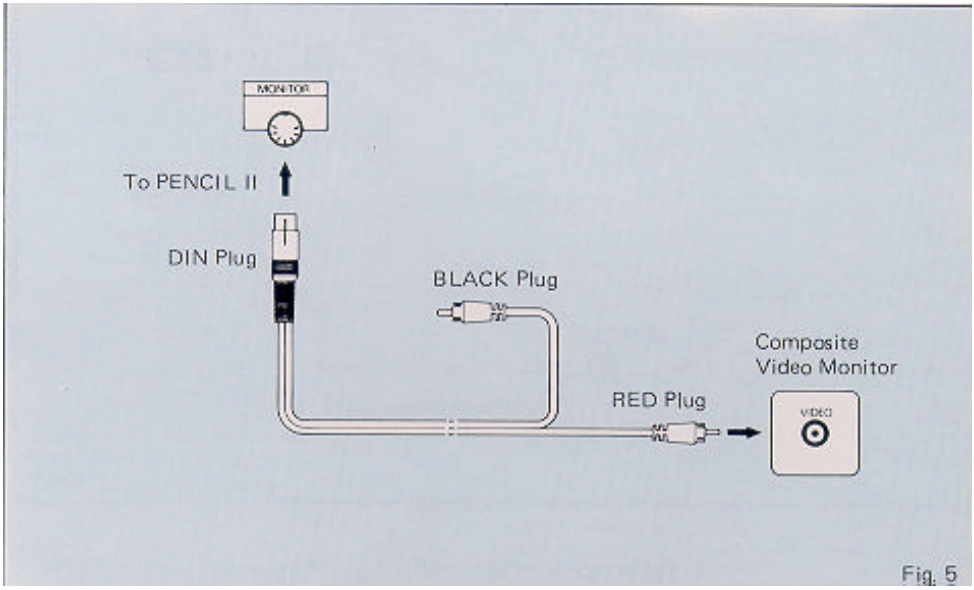


Fig. 5

Conecting to a Data assette Recorder

1. You can save your programs onto magnetic tapes by using a PENCIL Data Cassette Recorder (PEN-511, optional) and load them back to your PENCIL II for later use. However, it is possible to use any regular mono cassette tape recorder. One with a tape counter is preferable so you can locate your programs easily.

2. Normal cassette tapes can be used but low noise cassette tapes are recommended for better recording quality.

3. Use the Cassette Interface Cable for connections as shown in fig. 6.

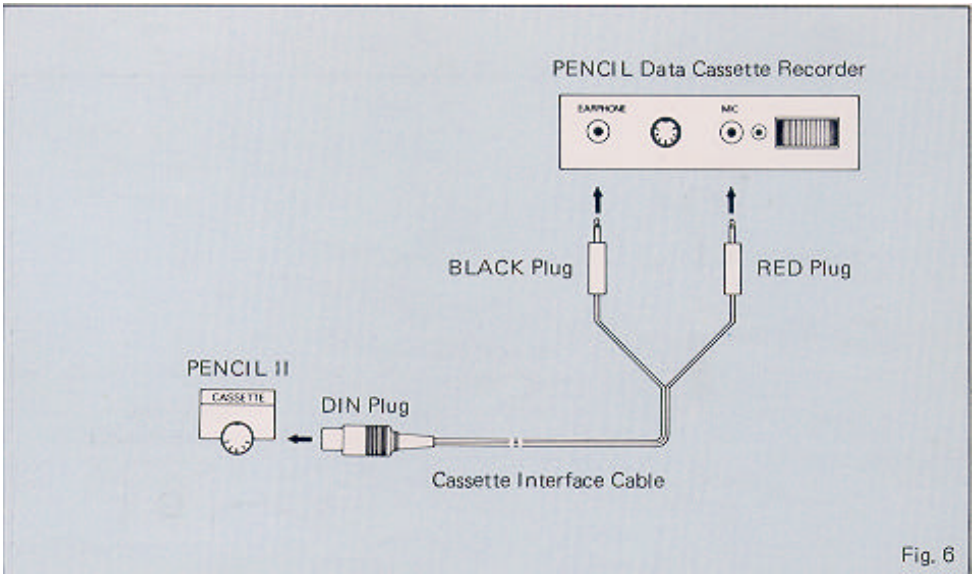


Fig. 6

4. Refer to separate Instruction Manual included in PENCIL Data Cassette Recorder and the SD-BASIC Reference Manual for detailed loading and saving instructions.

5. Set the volume control to an appropriate level which PENCIL II will accept. There is a load level indicator in PENCIL Data Cassette Recorder to indicate correct output level. When using a regular mono cassette tape recorder, this can only be achieved by a trial and error method. You can load the program into PENCIL 11 and adjust the volume level gradually at each trial until the program is successfully loaded. For reliability, we highly recommend using PENCI L Data Cassette Recorder.

5 HOW TO START

1. Make sure all connections are properly made before starting,
2. Switch on the TV set or monitor first and then your PENCIL II. The Power Indicator will light up.
3. If you are using a composite video monitor, you should see the "Greeting" picture on the screen within a few seconds as shown in fig. 7.

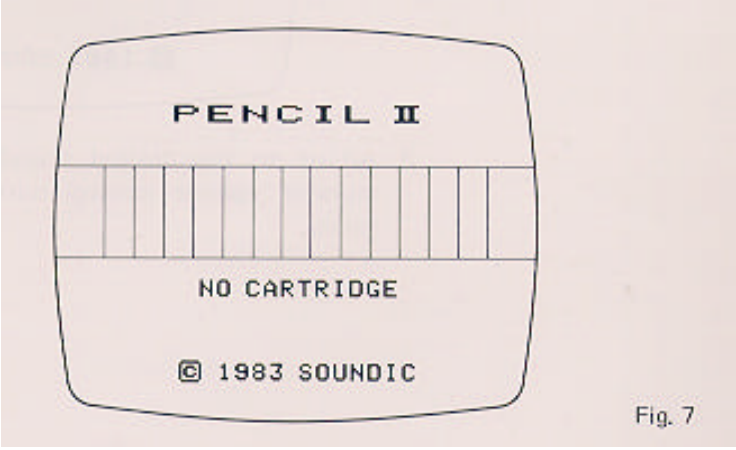


Fig. 7

4. If you are using a regular TV set, tune it to the proper channel until the "Greeting" picture shows its maximum clarity. Refer to the label at the bottom of PENCIL II to determine the proper channel.
5. You may find an optional Channel Switch at the bottom of certain versions of PENCIL II. The purpose of this switch is to ensure that your TV set gives the best video display without interference from the strong TV broadcasting signals in some areas. If your PENCIL II has such a switch, set it to either A or B position and tune your TV set to the corresponding channel as shown on the bottom level.
6. Switch off the AFT switch on the TV set, if any, when adjusting the channel frequency.

7. Using the colour blocks of the "Greeting" picture as a reference, tune your TV set or monitor to the desired colour, brightness and contrast level as shown in fig. 8.

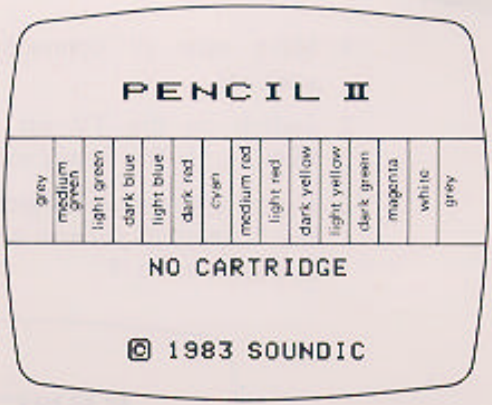


Fig. 8

8. Adjust to the desired sound level by using the TV or monitor volume control during programming or playing . games.

6 START WITH PROGRAM CARTRIDGE

1. Switch off your PENCIL II.

WARNING

To avoid damage to the PENCIL II, always switch off your PENCIL II before inserting or removing cartridges.

2. Insert the program cartridge into the cartridge slot of PENCIL II with the label facing the keyboard as shown in.

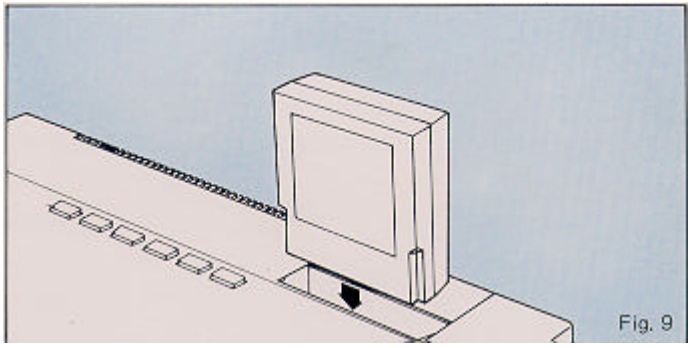


Fig. 9

3. Switch on the PENCIL II.

4. The "Greeting" picture (fig. 10) will be shown within a few seconds.

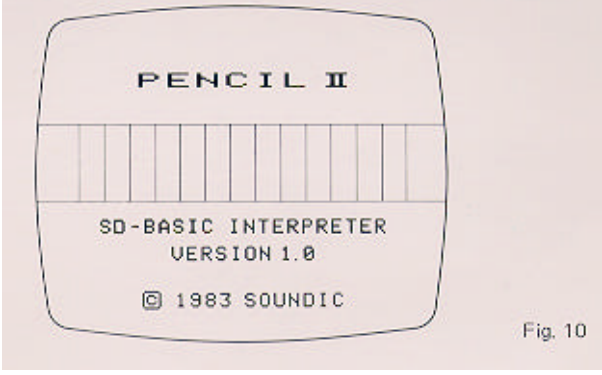


Fig. 10

5. The program is ready.

NOTE

Refer to the corresponding program manual for detailed commands and information.

If you are using other peripherals such as parallel printers, disk drive units, etc., refer to separate instruction manuals.

7 CARE AND MAINTENANCE

:

- Ensure the power supply in your house corresponds to the requirements stipulated on the label of the Power Supply Unit (PEN-902).
- Always switch off the PENCIL II before inserting or removing cartridges/peripherals, or when not in use.
- To disconnect the AC supply, never pull by the cord. Pull out by grasping the AC plug.
- Do not place the PENCIL II under direct sunlight; near stoves or heaters. Place in a well ventilated place.
- Do not place the PENCIL II in a damp or dusty surrounding.
- Dg not spill water or any liquid onto the PENCIL II.
- Do not put heavy load on top of the PENCIL II.
- Do not place the PENCIL II near appliances having strong magnetic field.
- Avoid severe shock or vibration to the PENCIL II while in use or during storage.
- Remove the cartridge from the cartridge slot when not in use.
- Clean the PENCIL II with a soft and dry cloth. Never clean it with solvents such as benzene, thinner or volatile chemicals as these may cause permanent damage to the computer.
- If the PENCIL II malfunctions, do not try to repair yourself. Switch off the power and consult your dealer for repairing services.

PART II

SD-BASIC REFERENCE MANUAL

Specifications and information of this manual are subject to change without notice.

All efforts have been made to supply accurate and complete information in this manual. Soundic Electronics Limited assumes no responsibility for any errors in this manual of their consequences.

PENCIL™ is a trademark of Soundic Electronics Limited

First Edition 1983
Printed in Hong Kong

PREFACE

This manual is intended to give the PENCIL II user an overview on the SD-BASIC language and its implementation.

This manual is not intended to teach BASIC. Users-who find difficulties in understanding this manual should read some relevant books on elementary BASIC.

Those who are already familiar with BASIC would know that BASIC interpreter in one system is different from the other. Therefore, it is recommended that ALL users should read through this manual in order to have a clear understanding of SD-BASIC.

CONTENTS

1 INTRODUCTION

1.1 Fundamental ideas

Before discussing SD-BASIC, followings are introduced to users who do not know how a computer works. For those who have experience on computers may skip the rest of this chapter.

1.2 What is a computer?

A computer is a device which performs various operations based on user-supplied instructions.

Generally, a computer consists of:

1.2.1 CENTRAL PROCESSING UNIT

This is the "brain" of the computer. It can perform both mathematical and logical operations. Note .that the word "brain" is not meant literally because the Central Processing Unit does not think by itself.

1.2.2 MEMORY

Storage area in the computer and where programs or data are stored.

1.2.3 INPUT OR OUTPUT DEVICES

These are the bridges keyboard (input), printer (output), etc. For example, programs or between the computer and users which include data are sent to the computer through the keyboard and the result is printed out from the printer.

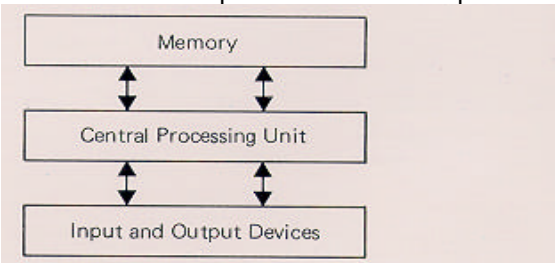


Fig. 1.2 System Configuration

1.3 What is BASIC

BASIC, Beginners' All-purpose Symbolic Instruction Code, is one of the most useful programming languages. BASIC uses simple English vocabulary. It uses general logical rules and can manipulate mathematical operations. BASIC is one of the most popular programming languages and all personal computer users should get themselves familiar with it.

2 HOW DOES SD-BASIC WORK IN PENCIL II

SD-BASIC is a BASIC interpreter available in cartridge form. Please follow the steps below to set up the system.

2.1 Procedures to set up SD-BASIC in PENCIL II

- To set up PENCIL II, please refer to the Operation Manual.
- Switch off PENCIL II computer console and connected peripherals.
- Insert the SD-BASIC cartridge into the Program Cartridge Slot.

  WHEN YOU ARE INSERTING/REMOVING THE CARTRIDGE, MAKE SURE THE POWER SWICH IS AT "OFF" POSITION.

- Switch on PENCIL II.
- Press the RESET button once. The SD-BASIC"-.Greeting" picture will be displayed on the screen as shown in fig.- 2.1. If it does not appear, check that all connections are properly made and the cartridge is properly inserted.

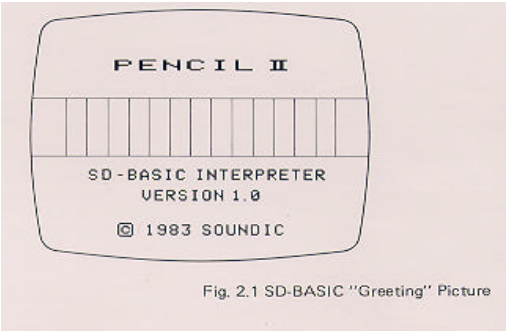


Fig. 2.1 SD-BASIC "Greeting" Picture

- For connection to other accessories, such as data cassette recorders, printers, memory packs, etc., please refer to the Operation Manual or the corresponding manuals.

3 THE PENCIL II KEYBOARD

PENCIL II keyboard is much more sophisticated than a typewriter because most keys are multifunctional, and its operation can be easily familiarized.

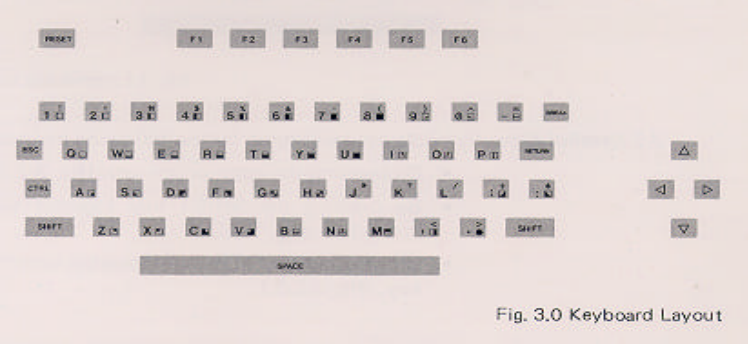


Fig. 3.0 Keyboard Layout

3.1.1 26 LOWER CASE ALPHABETIC KEYS

Fig. 3.0 Keyboard Layout

Lower case alphabets can be entered by pressing the appropriate character key.

3.1.2 26 UPPER CASE ALPHABETIC KEYS

Upper case alphabets can be entered by pressing the SHIFT key together with the appropriate character key.

Pressing the ESC key together with the SHIFT key changes the keyboard mode to upper case alphabets only; in such case, operation of other keys are unchanged. The cursor will be changed to ■ . Pressing these two keys again will release this keyboard mode and the cursor will be restored to its normal form ▣.

3.1.3 NUMERIC KEYS

Numbers can be entered by pressing the appropriate numeric key.

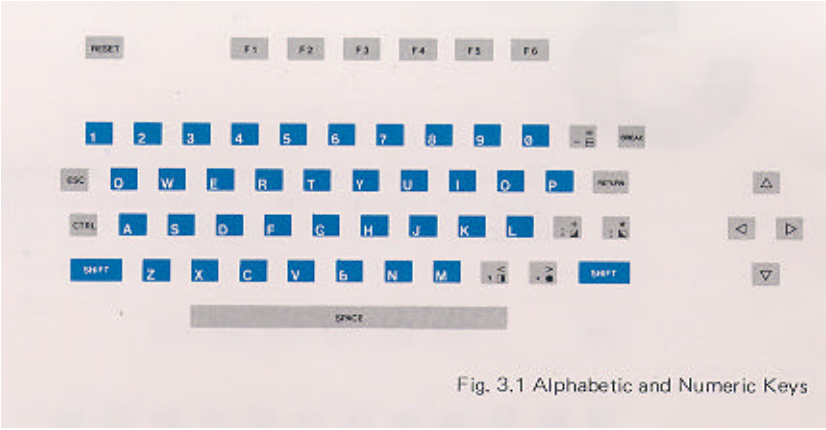


Fig. 3.1 Alphabetic and Numeric Keys

Symbols can be entered by the following methods:

- By pressing the appropriate key. (fig. *3.2 1)*
- By pressing the SHIFT key together with the appropriate key. (fig. *3.2.2.)*
- By pressing the CTR L key together with the appropriate key. (fig. *3.2.3.)*

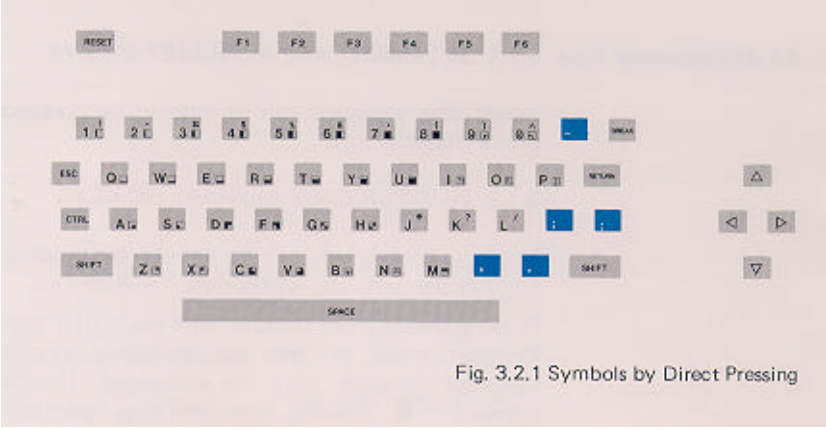


Fig. 3.2.1 Symbols by Direct Pressing

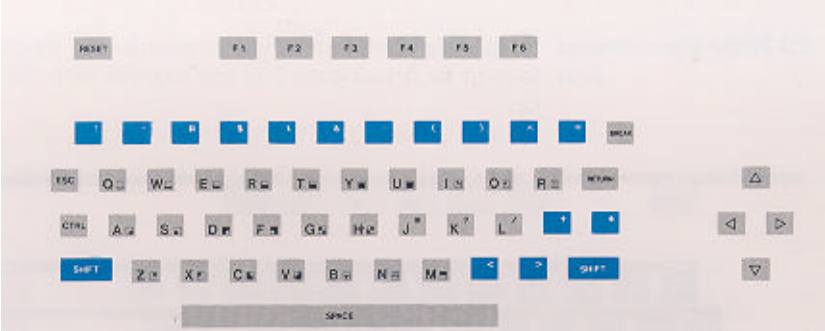


Fig. 3.2.2 Symbols by Pressing Together with the SHIFT Key

Fig. 3.2.3 Symbols by Pressing Together with the CTRL Key

Semi-graphics can be entered by pressing the CTRL key together with the appropriate key.

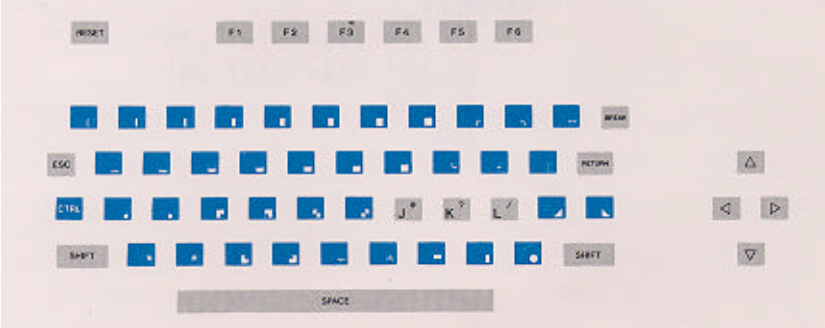


Fig. 3.3 Semi-graphic Characters

Frequently used commands, keywords and functions can be entered by pressing the ESC key together with the appropriate key.

keys

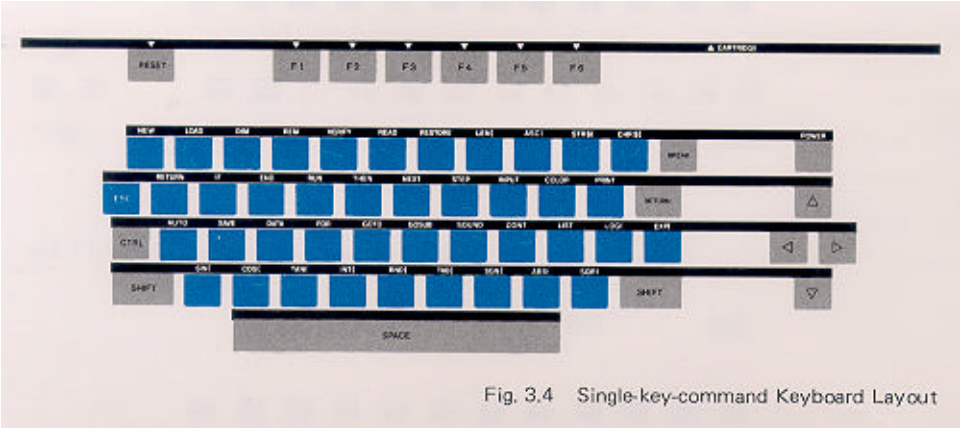


Fig. 3.4 Single-key-command Keyboard Layout

6 user-defined function keys, each capable of performing 2 user-defined functions, are available.

2 variable names are assigned to each of the F1, ..., F6 keys. The first one is *fx$*, where x is 1 to *6*, and can be recalled by pressing the corresponding function key. The second one is Fx$, and can be recalled by pressing the SHIFT key together with the corresponding function key.

Functions can be stored by using LET or implied LET statement commands. However, the string is restricted to 8 characters maximum.

EXAMPLE

```
10 LET f1$=”LIST ,30"
20 LET F1$="LIST 20”
30 END RUN
```

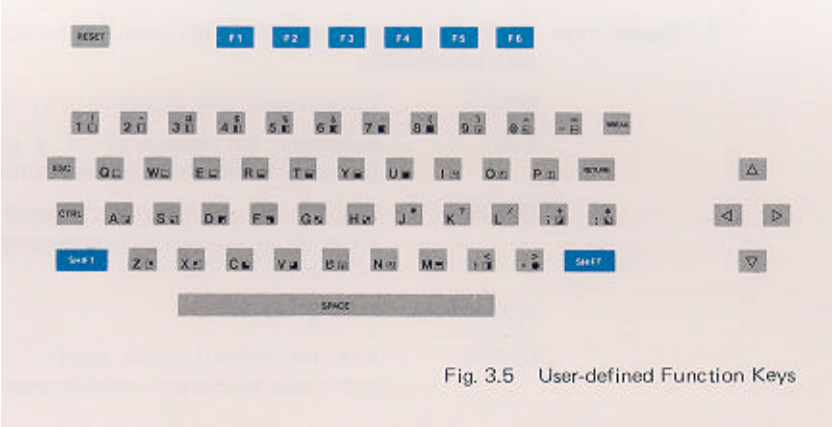


Fig. 3.5 User-defined Function Keys

	◀	Moves the cursor backward.
Shift	◀	Deletes the character at the cursor position.
	▶	Moves the cursor forward.
Shift	▶	Inserts a space to the right of the cursor.
	▲	Moves the cursor upward.
	▼	Moves the cursor downward.

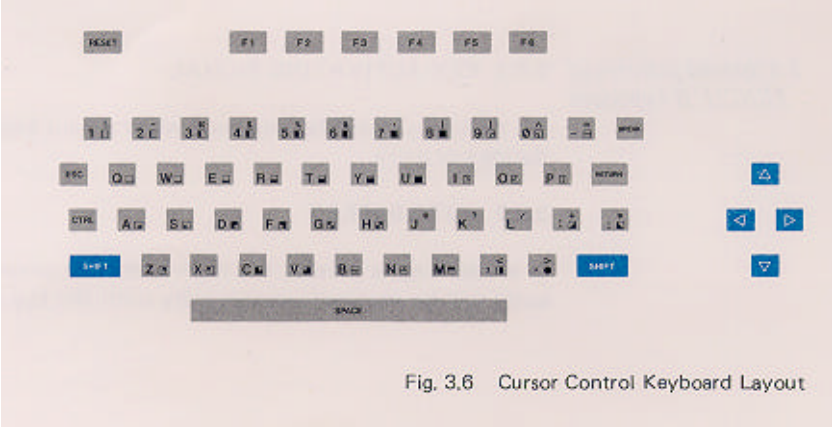


Fig. 3.6 Cursor Control Keyboard Layout

Apart from the above-mentioned keys, following special keys are also available:

SHIFT	Enters Upper case alphabets or symbols.
ESC	Suspends the execution of a command or program. Enters the single-key-commands.
BREAK	Stops the execution of a command or program. Exits from the automatic generation of line numbers in editing mode.
RETURN	Ends a line entry.
SPACE	Fills the current cursor position with a space and moves the cursor one character to the right.

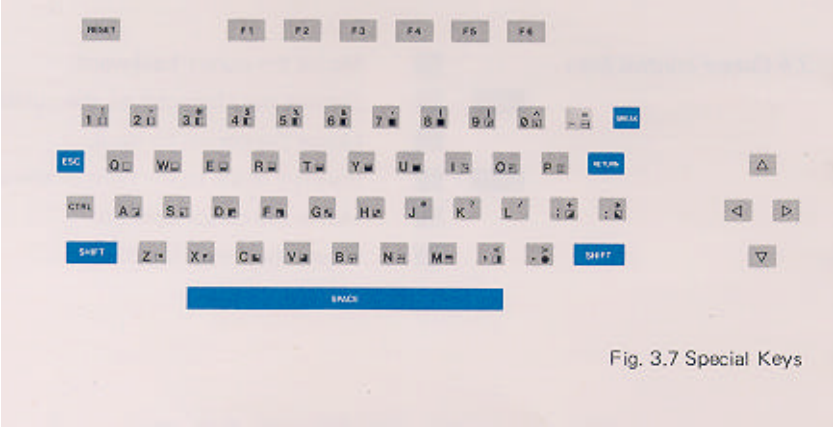


Fig. 3.7 Special Keys

3.8.1 KEY-ACTIVATING SIGNAL

An audio beep will be generated when a valid key is pressed during key entry.

3.8.2 AUTO-REPEAT

If a valid key is pressed for more than one second, the key entry will be repeated automatically until the key is released.

4 DATA TYPES

4.1.1 NUMERIC VALUES

Numeric values used in SD-BASIC are in two formats, fixed-point and floating point notations. They can either be positive or negative, but must be in the range of 10^{-38} to 10^{+38}

4.1.1.1 Fixed-point Notation

INPUT FORMAT Up to 6 significant digits can be entered in fixed-point notation.

OUTPUT FORMAT Numeric output will be in fixed-point notation if in the range of 0.1 to 999999 .

EXAMPLE

```
10.24
-82315
```

4.1.1.2 Floating Point Notation

INPUT FORMAT All numeric values can be entered in floating point notation.

OUTPUT FORMAT Numeric values which cannot be displayed in fixed-point notation are automatically displayed in floating point notation.

EXAMPLE

```
0.12E+12 ( 0.12E + 12 means 0.12 * 10^+12)
-0. 23E-18 (-0.23E—18 means -0.23 * 10^-48)
```

4.1.2 STRING VALUES

A string is a sequence of characters. In SD-BASIC a string can have a maximum of 32 characters.

INPUT FORMAT String values must be enclosed in quotation marks ("..." or '...') in line statements. However, when the computer asks for data entry (prompts with a ?), the string is entered directly through the keyboard without quotation marks.

OUTPUT FORMAT String values are printed without quotation marks.

EXAMPLE

```
10 A$ = 'HELLO! "
20 INPUT B$
30 PRINT AS; BS
RUN
?HOW ARE YOU?
HELLO! HOW ARE YOU?
```

As the word "variable" implies, a variable is a value which can be changed during program execution. Each variable must be named for identification.

1 or 2 alphanumeric characters, the first one must be an alphabet, can be used as variable names.

As variables may be numeric or string, a variable name ends with a $ sign denotes a string variable.

Note that the following variable names are reserved for user-defined function keys and cannot be Used as variable names:

```
f1$, f2$, f3$, f4$, f5$, f6$

F1$, F2$, F3$, F4$, F5$, F6$
```

EXAMPLE

```
10 A1 = 2.3
20 INPUT A
30 B2 = A + A1
40 PRINT B2
50 A1$ = "ABC
60 PRINT A1$
```

Where A1, A, B2 are numeric variable names and A1$ is a string variable name.

5 OPERATORS

Arithmetic expressions are evaluated according to the operators Used. Each operator is governed according to normal mathematical precedence.

The arithmetic operators, in the order of their precedence (from highest to lowest), are listed below:

Operator	Description
(, **)**	Parenthesis
^	Exponentiation
***** , **/**	Multiplication and Division
+, **-**	Addition and Subtraction

```
EXAMPLE
    10 A = 5 - ( 2 - 1 )
    20 B = 32
    30 C = 3 *4 / 2
    40 D = 5 - 2 - 1
    50 PRINT A
    60 PRINT B
    70 PRINT C
    80 PRINT D
    RUN
      4
      9
      6
      2
```

In addition to performing arithmetic calculations, operator + can be used to merge 2 strings. The result is limited to 32 characters.

```
EXAMPLE
    10 A$ = “GOOD ”
    20 BS = “MORNING”
    30 C$ = AS + B$
    40 PRINT C$
    RUN
    GOOD MORNING
```

Relational operators are used to compare the arithmetic and string expressions on both sides of the operator.

The relational operators and their definitions are listed below:

Operator	Description
=	Equal
< >	Not Equal
<	Less than
>	Greater than
< =	Less than or Equal
> =	Greater than or Equal

These operators are used to perform logical decisions in the IF statements. For details, please refer to "**9.1 *The IF... THEN. - and IF...GOTO Statements***".

```
EXAMPLE
    10 INPUT A, BS
    20 IF A )= 0 THEN PRINT A
    30 IF BS ( IIM'' THEN PRINT i3S
    40 IF A - O THEN 60
    50 GOTO 10
    60 END
```

Logical operators are used in the IF...THEN statements to connect two or more relational expressions to decide the directions of the program.

Logical operators have the lowest hierarchy among the operators and is performed after arithmetic and relational operations,

The logical operator, in the order of their precedence (from highest to lowest}, are listed below:

Operator	Description
NOT	Logical Negative
AND	Logical And
OR	Logical Or

EXAMPLE

```
10 INPUT A, B
20 IF NOT A = 0 THEN PRINT
   “A NOT EQUAL ZEROS”
30 IF A > B AND B > 0 THEN 60
40 IF A = 0 OR B = 0 THEN 80
5O GOTO 10
60 PRINT “A > B > 0”
70 GOTO 10
80 END
```

6 EDITOR

Editors are used to edit program line statements.

6.1.1 TO INPUT A PROGRAM

You can enter your program lines when the prompt sign OK appears on the screen. Always start with a line number. Program lines can be entered in any order since the computer will arrange the sequence accordingly.

EXAMPLE

```
10 A = 2
20 INPUT B
30 C = A + B
40 PRINT C
50 END
```

6.1.2 TO DELETE A LINE

Entering a line number only will delete the specified line from the program.

EXAMPLE **To delete line** 20

```
20
LIST
10 A = 2
30 C = A+B
40 PRINT C
50 END
```

6.1.3 TO INSERT A LINE

A new line can always be inserted into the program provided it starts with a valid line number.

EXAMPLE **To insert line 15**

```
15 B = 4
LIST
```

```
10 A = 2
15 B = 4
30 C = A+B
40 PRINT C
50 END
```

6.1.4 TO MODIFY A LINE

A program line can always be modified by entering the line number followed by a new statement.

EXAMPLE **To modify line** 15

```
15 INPUT B
LIST
10 A = 2
15 INPUT B
30 C = A+B
40 PRINT C
50 END
```

SD-BASIC also allows you to modify any program line that is displayed on the screen without retyping the whole line all over again.

Move the cursor, using the cursor control keys, to the position where a modification is needed. Replace by typing the new information and followed by pressing the RETURN key. Use the SHIFT key together with cursor control keys to delete or insert a character as described in *"**3.6 Cursor control keys**"*.

7 COMMANDS

SD-BASIC commands are used to control operations of the SD-BASIC interpreter.

This chapter will discuss the following SD-BASIC commands:

- **NEW**
- **AUTO**
- **LIST**
- **LLIST**
- **RUN**
- **CONT**

FORMAT **NEW**

PURPOSE To erase the current program in memory.

COMMENTS Care should be taken when using this command because it will erase your program in memory. Save your program before using this command.

EXAMPLE

```
10 A = 2
20 PRINT A
30 END
NEW
OK
LIST
```

No program is displayed because the program has been erased by the NEW command.

FORMAT **AUTO**
AUTO line number
AUTO line number, increment

PURPOSE To generate line numbers automatically after each carriage return in editing mode.

COMMENTS The first value specifies the starting line number. The second one specifies the incremental interval. The default for both values is 10.

To stop generating line numbers, press BREAK key.

EXAMPLE

```
AUTO            Generates line numbers
                10, 20, ...

AUT0 100        Generates line numbers
                100, 110, 120, ...

AUTO 100,20     Generates line numbers
                100, 120, 140, ....
```

FORMAT **LIST**
LIST ***line number***
LIST ***line number,***
LIST ***line number, line number***
LIST , *line number*

PURPOSE To instruct the computer to display the program onto the screen.

COMMENTS It is useful to list the program for checking any obvious errors. Pressing the ESC key will suspend the program being scrolled out onto the screen. Pressing any other key will resume the listing process. The BREAK key can be used to terminate the listing process.

EXAMPLE

```
LIST            Displays the entire program
                currently in memory.
LIST 100        Displays line 100.
LIST 100,       Displays from line 100
                to the end of the program.
LIST 10O,200    Displays from line
                100 to 200.
LIST , 200      Displays from the
                Beginning to line 200.
```

FORMAT **LLIST**
LLIST line number
LLIST line number,
LLIST line number, line number
LLIST, line number

PURPOSE To give a hard copy of a program on the printer.

COMMENTS Make sure the printer has been set up properly -. before using this command. For setting up of the printer, please refer to the Operation Manual.

LLIST is identical to LIST except that the output goes to the printer instead of the screen.

EXAMPLE

```
    LLIST           Prints the entire program
                    currently in memory.

    LLIST 100       Prints line 100.
    LLIST 100,      Prints from line l00 to
                    the end of the program.
    LLIST 100,200   Prints from line 100
                    to 200.
    LLIST , 200     Prints from the beginning
                    to line 200.
```

FORMAT **RUN**

PURPOSE To execute the program.

COMMENTS Pressing the ESC key will suspend program execution. Pressing any other key will resume the program execution. The BREAK key can be used to stop the program execution and return to command mode,

EXAMPLE

```
    10 PRINT "GOOD AFTERNOON"
    20 GOTO 10
    RUN
```

FORMAT **CONT**

PURPOSE To continue program execution which has been stopped by a **STOP** statement or the BREAK key.

COMMENTS If the program execution is stopped due to a program error, this command will be ineffective.

EXAMPLE

```
10 INPUT A
20 STOP
30 G0T0 10
RUN
?0
STOP AT LINE 20

OK

CONT
```

When this program is executed, it will stop at line 20 after each entry of a value A.

8 SIMPLE STATEMENTS

A SD-BASIC program consists of a series of instructions which are executed by the computer to perform a certain task. Each instruction is known as a statement.

A statement must start with a line number to indicate its sequence.

SD-BASIC statements have many different keywords and built-in functions, which will be discussed in the following chapters.

In this chapter, we will discuss the following SD-BASIC simple statements:

- LET
- GOTO
- GOSUB and RETURN
- READ DATA and RESTORE
- REM
- CALL
- POKE
- DIM

FORMAT ***LET variable*** = ***expression***

PURPOSE To assign value to a variable.

COMMENTS The **=** operator should be interpreted as "take the value of". Thus the statement means **LET** the variable takes the value of the expression. Variables or expressions on both sides of the = operator must be of the same type, i.e. numeric or string. As **LET** statement is often used, SD-BASIC interpreter allows the user to omit the keyword **LET**.

EXAMPLE

```
    10 LET A = 12.3
    20 LET B$ = "GOOD MORNIG”
```

```
30 A = 12.3
40 B$ = "GOOD MORNING"
```

Lines 30 and 40 mean exactly the same as lines 10 and 20.

FORMAT **GOTO** ***line number***

PURPOSE To transfer program execution unconditionally to the specified program line.

COMMENTS **GOTO** statement overrides the normal program execution sequence. The specified line number is usually not the next line in sequence, thus the execution sequence can be altered at user's choice. However, care should be taken to avoid going to any statement inside another **FOR** loop; doing so may result program execution errors.

EXAMPLE

```
10 A = 0
20 A = A+1
30 PRINT A
40 IF A = 100 G0T0 10
50 GOT0 20
```

This program will result in a dead loop. Use the BREAK key to stop its execution.

FORMAT **GOSUB** ***line number***
RETURN

PURPOSE A sequence of statements with a **RETURN** as its last statement is known as a subroutine. The **GOSUB** statement is used to call the subroutine. After execution of the subroutine, the statement following the **GOSUB** statement will be executed.

COMMENTS Subroutines are very useful because it is often found that segments of a program are repeatedly used in a program. With the use of subroutines, the required program memory size can be reduced.

EXAMPLE

```
10 K = 2
20 GOSUB 60
30 K = 5
40 GOSUB 60
50 END
```

```
60 REM SUBROUTINES
70 A - K * K
80 PRINT " THE SQUARE OF "¡K;" =";A
90 RETURN
```

The main program is lines 10 through 50, and the subroutine is placed outside the main program. Without the **END** statement in line 50, the program will continue to execute after line 40 and fatal error will occur when the **RETURN** statement in line 90 is encountered.

FORMAT **DATA** ***val1, val2, val3, ..., valN***
READ ***variable1, variable2, variable3, ..., variableN***
RESTORE

PURPOSE Values supplied by the **DATA** statements will be assigned to the variables in a **READ** statement. The **RESTORE** statement is used to reset the data pointer to the first value in the **DATA** statements.

COMMENTS The **READ** statement reads the values in the **DATA** statements sequentially. **DATA** statements need not be grouped together and can be placed anywhere in the program.

If excess data are supplied, the extra data :are ignored. If insufficient data are supplied, fatal error will occur.

The SD-BASIC contains a data pointer that keeps track of the values available in **DATA** statements. When a value is read, the data pointer will advance to the next value. The **RESTORE** statement resets this pointer to the first value in the program and allows repetitive use of valves in **DATA** statements.

EXAMPLE

```
10 DATA 1,2,3,4
20 FOR J = 1 TO 5
30 READ A
40 PRINT A
50 NEXT J
60 DATA 5,6,7,8
70 RESTORE
8O FOR J = 1 TO 4
90 READ A, B
100 PRINT A, B
110 NEXT J
120 END
RUN
```

FORMAT **REM** ***any character string***

PURPOSE To make explanatory notes and messages for easy understanding and reference to the program.

COMMENTS As the word suggests, this is only a remark and the message in this line will not be executed.

EXAMPLE

```
10 REM THIS lS A REMARK STATEMENT
20 REM PROGRAMMED BY DAVID
30 PRINT “OK”
40 END
```

FORMAT **CALL** ***address***

PURPOSE To execute a subroutine written in machine code.

COMMENTS It is more efficient to run a machine code program. This statement calls the subroutine which starts at the specified memory address and ends with a "RTS" machine code instruction.

This statement is not recommended for users without good knowledge of the Z80 machine code instructions.

EXAMPLE

```
10 CALL 32770
20 END
```

FORMAT **POKE** ***address, value***

PURPOSE To write a value into a specified memory location.

COMMENTS Both the address and the value are positive integers. The maximum address is 65535 and the maximum value is 255.

EXAMPLE

```
10 POKE 29439, 144
20 END
```

FORMAT **DIM** ***variable-name (N1), ..., variable-name (Nn)***
DIM ***variable-name (N1, N2), ..., variable-name (Nm, Nn)***

PURPOSE To define a one-dimensional or two-dimensional array.

COMMENTS Array names can be same as variable names. However, the variable name can have 1 character only. PENCIL II will identify an array by the subscript after the name.

EXAMPLE

```
10 DIM A(2,3)
20 A = 3.5
30 FOR J = I T0 2
40 FOR K ~ 1 T0 3
50 A(J,K) = 3 * (J-1) + K
ó0 PRINT A(J,K)
70 NEXT K
80 NEXT J
90 PRINT A
100 END
```

8 SD-BASIC CONTROL AND LOOPING STATEMENTS

In this chapter, we will introduce the following control and looping statements

- IF... THEN and IF ... GOTO
- FOR ... TO ... STEP and NEXT

FORMAT **IF** ***logical expression*** **THEN** ***statement***
IF ***logical expression*** **THEN** ***line number***
IF ***logical expression*** **GOTO** ***line number***

PURPOSE To direct the program flow based on a logical decision.

COMMENTS The logical expression will be evaluated when this statement is encountered. If the result is true, either the statement will be executed or the program will go to the line number specified. If the result is false, 'he rest of the' statement will be ignored.

EXAMPLE

```
10 A = O
20 A = A
30 IF A < 6 THEN PRINT “#”;
40 IF A < 6 THEN GOTO 20
50 REM
60 IF A < 11 THEN PRINT “*”;
70 IF A < 11 THEN 20
80 REM
90 IF A < 16 THEN GOSUB 120
100 IF A < 16 GOTO 20
110 END
120 PRINT “$”;
130 RETURN
RUN
#####*****$$$$$
```

FORMAT **FOR** ***index*** = ***exp1*** **TO** ***exp2*** **STEP** ***exp3***
NEXT ***index***

PURPOSE To execute a sequence of statements a number of times.

COMMENTS A loop is a sequence of statements within the **FOR** and the corresponding **NEXT** statements

index is the control or counting variable of the loop.
exp1 is the initial value of the index.
exp2 is the terminating index value for the loop.
exp3 is the step value added to the index for each execution of the loop.

Upon execution of the **FOR** statement index starts from exp1 and the loop is executed. When each **NEXT** statement is encountered, the value of ***exp3*** is added to index. The new value of index is then compared with the terminating limit ***exp2***. The loop will be executed again if index has not reached the terminating value.

Note that ***exp3*** may be a negative number so that index is decremented after each execution count. Stepping value will be 1 if **STEP** ***exp3*** is omitted.

```
EXAMPLE 1 Single Loop Program
    10 FOR K = 1 TO 10 STEP 2
    20 PRINT K
    30 NEXT K
    40 END

EXAMPLE 2 Nested Loop Program
    10 FOR I = I T0 2
    20 FOR J = 1 T0 3
    30 PRINT I, J, I*J
    40 NEXT J
    50 NEXT I
    60 END
```

There are two loops in this example, I and J-loop. Loops can be nested, one within the other. The inner loop (J-loop) is completed before returning to the outer loop (I-loop).

10 INPUT AND OUTPUT

In this chapter, we will discuss the following input and output statements:

- INPUT
- PRINT
- LPRINT
- CLS
- COLOR
- SOUND
- PRINT#
- INPUT #

FORMAT **INPUT variable1, variable2, ..., variableN**

PURPOSE To enter data through the keyboard during program execution.

COMMENTS When an **INPUT** statement is executed, the computer displays **?** and waits for the user to enter data through the keyboard. Each data is, entered using the RETURN key. Another **?** will be displayed if more data are required by the **INPUT** statement.

```
EXAMPLE 1
    10 INPUT A, B
    20 PRINT A+B
    RUN
    ?4
    ?5
    9
EXAMPLE 2
    10 INPUT A$
    20 INPUT B$
    30 PRINT A$+B$
    RUN
    ?GOOD
    ? EVENING
    GOOD EVENING
```

FORMAT **PRINT** ***val1 del1 val2 del2 ... valN delN***

PURPOSE To display the values of variables or expressions.

COMMENTS The ***val1, val2, ..., valN*** can either be values, variables or expressions. They can be numeric or string. The ***del1, del2, ... delN*** are delimiters and can either be commas or semicolons. If it is a semicolon, the display will leave no space between values.

The last delimiter is optional. If omitted, the cursor will move to a new line.

EXAMPLE

```
10 A = 234
20 PRINT 0.1, A; 5. 6
30 B$ = "APRIL"
40 PRINT B$
RUN
0.1    234 5.6
APRIL
```

FORMAT **LPRINT** ***val1 del1 val2 del2... valN delN***

PURPOSE Same as **PRINT** statement except that the output goes to the printer instead of the screen.

COMMENTS Same as **PRINT** statement. When **LPRINT** is used, make sure the printer is properly set up and ready for printing. Otherwise, the computer will wait until the printer is ready.

EXAMPLE

```
10 A = 234
20 LPRINT A
30 B$ = "APRIL"
40 LPRINT B$
RUN
```

FORMAT **CLS**

PURPOSE To clear the screen and to bring the cursor back to home position.

COMMENTS Do not mix up this statement with the **NEW** command. **CLS** only clears the screen and your program is still in the memory.

EXAMPLE

```
10 CLS
20 PRINT "CLEARED"
30 END
```

FORMAT **COLOR** ***character-colour, background-colour***

PURPOSE To set the character and background colours

COMMENTS Please note-that the keyword is **COLOR** and not COLOUR. The colour codes are listed below:

Code	**Colour**
0.	Transparent
1.	Black
2.	Medium Green
3.	Light Green
4.	Dark Blue
5.	Light Blue
6.	Dark Red
7.	Cyan
8.	Medium Red
9.	Light Red
10.	Dark Yellow
11.	Light Yellow
12.	Dark Grcen
13.	Magenta
14.	Grey
15.	White

EXAMPLE

```
10 FOR I = 0 T0 15
20 FOR J = 0 T0 15
30 COLOR I, J
40 NEXT J
50 NEXT I
60 COLOR 1,2
```

FORMAT **SOUND** ***generator-number, frequency, attenuation-value, duration; … ; generator-number, frequency, attenuation-value, duration*** $

PURPOSE To generate sound.

COMMENTS The **SOUND** statement enables the PENCIL II to generate tones of independent frequencies, and amplitudes. Generator number 0 to 2 are used to generate musical tones for composing any piece of music. Generator number 3 is a noise-generator used to create special sound effects.

The frequencies for the tone-generator are limited between 110 and 20000 Hertz, and the frequency codes of the noise generator are limited to 7 as listed below:

FREQUENCY CODE	DESCRIPTION
0	Periodic Noise with High Frequency
1	Periodic Noise with Medium Frequency
2	Periodic Noise with Slow Frequency
3	Periodic Noise with Frequency of Tone Generator Number 2
4	White Noise with High Frequency
5	White Noise with Medium Frequency
6	White Noise with Slow Frequency
7	White Noise with Frequency of Tone Generator Number 2

The attenuation value ranges from 0 to 15. 0 gives the maximum output sound amplitude and 15 turns off the corresponding sound generator.

The sound duration can be 1 to 32 units and each unit is 0.1 second.

EXAMPLE

```
10 FOR J = 110 TO 10000 STEP 75
20 SOUND 0,J+30,1,1;1,J+50,4,1
30 NEXT J
```

FORMAT **PRINT#** ***filename:var1,var2,var3,...,varN***

PURPOSE To open a cassette file and store manipulated data onto the file.

COMMENTS Data most be either ..numeric or string variables

EXAMPLE

```
10 DIM A(10)
20 FOR 1=1 TO 10
30 A(1)=1*2
40 NEXT I
50 A=10.2
60 B$= “G00D MORNING”
70 PRINT “ON CASSETTE”
8O INPUT A$
9O PRINT# DATA1: A, B$, A(I), A(2), A(10)
100 PRINT "OFF CASSETTE"
110 END          -
```

FORMAT **INPUT#** ***filename: Var1, var2, var3, ..., varN***

PURPOSE To read in data from a specified cassette data file.

COMMENTS The data file specified must have the same format as the INPUT# statement or error will occur.

EXAMPLE

```
10 DIM A(10)
20 PRINT “ON CASSETTE'”
30 INPUT A$
40 INPUT# DATA1: A, B$, A(I), A(2), A(10)
50 PRINT “OFF CASSETTES”
60 PRINT A, B$, A(I), A(2), A(10)
70 END
```

11 SYSTEM CONTROL STATEMENTS

In this chapter, we will explain the following system control statements:

- STOP
- END

FORMAT **STOP**

PURPOSE To suspend program execution- and return. to . command mode.

COMMENTS **STOP** statements are used in program debugging. When **STOP** statement is encountered, the current line number will be displayed and the computer will return to the command mode. This is useful for tracing the program execution and for examining intermediate results. Program can be resumed using the **CONT** command.

EXAMPLE

```
10 A = 0
20 PRINT A
30 STOP
40 A = A + l
50 STOP
60 GOTO 20
RUN
0
STOP AT LINE 30
```

FORMAT **END**

PURPOSE To terminate the execution of the program.

COMMENTS This can be placed anywhere in the program Computer will return to command mode if this statement is

encountered or if there are no more statements to be executed.

```
EXAMPLE
    10 A = 12
    20 GOSUB 40
    30 END
    40 PRINT A * 12
    50 RETURN
    RUN
     144
```

12 NUMERIC FUNCTIONS

SD-BASIC supports a lot of mathematical functions as below:

- ABS
- SQR
- INT
- SGN
- RND
- TAB
- EXP
- LOG
- PEEK
- SIN
- COS
- TAN
- ATN

These functions can only be used in expressions. A function by itself cannot be a statement.

FORMAT **ABS(*x*)**

PURPOSE To obtain the absolute value of ***x***. The absolute value is the magnitude of the number regardless of its sign.

EXAMPLE

```
10 A = ABS(10.2}
20 B = ABS (-4. 2 )
30 PRINT A
40 PRINT B
RUN
 10.2
 4.2
```

FORMAT **SQR{x)**

PURPOSE To obtain the square root value of x, where x must be positive.

```
EXAMPLE
    10 A = SQR(16)
    20 PRINT A
    RUN
     4
```

FORMAT **INT(*x*)**

PURPOSE To obtain the greatest integral number less than or equal to ***x***.

```
EXAMPLE 1
    10 PRINT INT(2.01)
    RUN
     2
EXAMPLE 2
    10 PRINT INT(-l.9)
    RUN
    -2
```

FORMAT **SGN(*x*)**

PURPOSE To obtain the arithmetic sign of ***x***. The return value is 1.0 if ***x*** is positive and -1.0 if ***x*** is negative. If ***x*** is zero, the return value will be 0.

```
EXAMPLE
    10 PRINT SGN(4.2)
    20 PRINT SGN(0)
    30 PRINT SGN(-5.4)
    RUN
     1
     0
    -1
```

FORMAT **RND(0)**
PURPOSE To obtain a random value between 0 and 1.

```
EXAMPLE
    10 PRINT RND(0)
    RUN
    0.248723
```

FORMAT **TAB(*x*)**

PURPOSE To move the cursor by ***x*** character positions in PRINT/LPRINT statements.

```
EXAMPLE
    10 FOR I = l T0 5
    20 PRINT TAB(5+I); “*'.
    30 NEXT I
    RUN
          *
           *
             *
               *
                 *
```

FORMAT **EXP(*x*)**

PURPOSE To obtain the natural exponential value of x, i.e. $\mathbf{e}^{x}$, where **e** is 2.71828.

```
EXAMPLE
    10 A=EXP (2.3)
    20 PRINT A
    RUN
     9.97418
```

FORMAT **LOG(*x*)**

PURPOSE To obtain the natural logarithmic value of ***x*** where ***x*** must be greater than 0.

```
EXAMPLE
    10 A= LOG (9.97418)
    20 PRINT A
    RUN
     2.3
```

FORMAT **PEEK(*x*)**

PURPOSE To obtain the value stored at the specified memory location.

EXAMPLE

```
10 A=PEEK (32817)
20 PRINT A
RUN
 83
```

FORMAT **SIN (*x*)**

PUIRPOSE To obtain the sine value of ***x*** where ***x*** is in radian.

EXAMPLE

```
10 A=SIN (0.5)
20 PRINT A
RUN
 0.479425
```

FORMAT **COS(*x*)**

PURPOSE To obtain the cosine value of ***x*** where ***x*** is in radian.

EXAMPLE

```
10 A=C0S(0.5)
20 PRINT A
RUN
 0.877582
```

FORMAT **TAN(*x*)**

PURPOSE To obtain the tangent value of ***x*** where ***x*** is in radian.

EXAMPLE

```
10 A = TAN(1.0)
20 PRINT A
RUN
 1.5574
```

FORMAT **ATN(*x*)**

PURPOSE To obtain the arctangent value of ***x***.

EXAMPLE

```
10 A=ATN(4.5)
20 PRINT A
RUN
1.35212
```

13 STRING FUNCTIONS

SD-BASIC also supports following string functions:

LEN
LEFT$
RIGHT$
MID$
CHR$
STR$
VAL
ASC

Note that string results will be obtained on functions which end with $ signs.

FORMAT **LEN(*x$*)**

PURPOSE To obtain the number of characters in the string expression ***x$***.

EXAMPLE

```
10 Y$ = "SD-BASIC IN PENCIL II"
20 A = LEN(Y$)
30 PRINT A
RUN
 21
```

FORMAT **LEFT$(*x$,n*)**

PURPOSE To obtain the leftmost ***n*** characters in the string expression ***x$***.

EXAMPLE

```
10 Y$ = "G00D MORNING"
20 Z$ = LEFT$(Y$,4)
3C, PRINT Z$
RUN
GOOD
```

FORMAT **RIGHT$(*x$,n*)**

PURPOSE To obtain the rightmost ***n*** characters in the string expression ***x$***.

```
EXAMPLE
    10 Z$ = “G00D AFTERNOON”
    20 Y$ = RIGHT$(Z$,9)
    30 PRINT Y$
    RUN
    AFTERNOON
```

FORMAT **MID$(*x$,m,n*)**

PURPOSE To obtain a substring of ***n*** characters which starts from the ***m*** th character in the string expression ***x$***.

```
EXAMPLE
    10 Z$ = "HOW  ARE YOU”
    20 Y$ = MID$(Z$,5,3)
    30 PRINT Y$
    RUN
    ARE
```

FORMAT **CHR$(*n*)**

PURPOSE To obtain a single character whose ASCI I code is ***n***.

```
EXAMPLE
    10 Y$= CHR$(66) + CHR$(67)
    20 PRINT Y$
    RUN
    BC
```

FORMAT **STR$(*n*)**

PURPOSE To convert the numeric value ***n*** to its string form.

```
EXAMPLE
    10 YS = STR$(45)
    20 PRINT Y$
    RUN
     45
```

FORMAT **VAL(*x$*)**

PURPOSE To convert the string ***x$*** to its numeric form. Note that ***x$*** must be a numeric expression.

```
EXAMP LE
    10 Y$ = “42*2+3*3”
    30 A = VAL (Y$)
    40 PRINT A
    RUN
     93
```

FORMAT **ASC(*x$*)**

PURPOSE To obtain the ASCll value of the first character in string expression ***x$***.

```
EXAMPLE
    10 P$ =”'XYZ”
    20 A = ASC (P$) .
    30 PRINT A
    RUN
     88
```

14 CASSETTE COMMANDS

This chapter will explain the loading and saving of programs using the data cassette recorder. For setting up of the data cassette recorder, please refer to the installation procedure in the Operation Manual.

FORMAT **SAVE *filename***

PURPOSE To save the program in memory onto the cassette tape for future retrieval.

COMMENTS For future retrieval of a program, this command is used to save the program onto the cassette tape. It is also a good practice to verify the program after the **SAVE** command is carried out. The filename can have a maximum of 8 characters.

EXAMPLE

```
SAVE FINANCE
```

PROCEDURES

- Put the cassette tape into the recorder and rewind or forward to the desired position.
- Note the tape counter reading for future reference.
- Key in SAVE filename.
- Set the recorder to record mode and run for 10 seconds.
- Press the RETURN key to save the program. The prompt sign OK will appear on the screen after the program has been saved. Stop the recorder.
- Note the tape counter reading for future reference.
- Verify the program using the VERIFY command.

FORMAT **LOAD** ***filename***
LOAD

PURPOSE To load a saved program from cassette tape into the computer.

COMMENTS The computer will examine all saved programs on cassette tape sequentially until it finds the selected program. During this process, skipped program filenames will be displayed for your reference.

If no filename is specified in the **LOAD** command, the first program encountered will be loaded.

EXAMPLE

```
LOAD FINANCE
PHONE IS SKIPPED
FINANCE IS FOUND
```

PROCEDUR ES

- Put the cassette tape into the recorder and rewind or forward to the desired position.
- Key in the LOAD filename command.
- Set the recorder to play mode. After the program has been loaded the prompt sign OK will appear on the screen.
- Stop the recorder.

FORMAT **VERIFY** ***filename***

PURPOSE To verify the saved program against the program in memory.

COMMENTS It is recommended to verify the saved program immediately after each saving process. This is to ensure that the program is saved correctly onto the cassette tape.

EXAMPLE

```
VERIFY FINANCE
FINANCE IS FOUND
```

PROCEDURES Same as the LOAD command.

15 CALCULATOR MODE

SD-BASIC can be operated as a calculator.

```
EXAMPLE 1
    PRINT 2*4 +25
     33

EXAMPLE 2

    P = 3.14

    OK

    PRINT P*2
     6.28

    OK

    PRINT P * 3
    9. 4 2
```

In this example, you first assign a value to P and use this variable in subsequent calculations.

A QUICK REFERENCE GUIDE

A.1 *Arithmetic operators*

+	Addition
-	Subtracion
*****	Multiplication
/	Division
^	Exponentiation

A.2 *Relational operators*

=	Equal
<>	Not Equal
<	Less than
>	Greater than
< =	Less than or Equal
> =	Greater than or Equal

A.3 *Logical operators*

NOT	Logical Negative
AND	Logical And
OR	Logical Or

A.4 *Commands*

AUTO
CONT
LIST
LLIST
LOAD
NEW
RUN
SAVE
VERIFY

A.5 *Keywords*

CALL
CLS
COLOR
DATA
DIM
END
FOR...TO...STEP
GOTO

GOSUB
IF...THEN
INPUT
INPUT #
LET
LPRINT
L PRINT
NEXT
POKE
PRINT
PRINT #
READ
REM
RESTORE
RETURN
SOUND
STOP

A 6 ***Numeric functions***

ABS
EXP
INT
LOG
PEEK
RND(0)
SGN
SQR
TAB
SIN
COS
TAN
ATN

A.7 ***String functions***

ASC
CHR$
LEFT$
LEN
MID$
RIGHT$
STR$
VAL

B ERROR CODE MESSAGE

CN	Cannot Continue
DE	Dimension
DZ	Divide by Zero
FC	Illegal Function Call
FN	FOR...NEXT
I0	Cassette I/0
OD	Out of Data
OM	Out of Memory
OR	Out of Range
OV	Overflow
RT	**RETURN** without **GOSUB**
SY	Syntax
TL	Too Long
TM	Type Mismatch
UL	Undefined Line Number
UN	Unbalanced Parentheses / Quotes
IN	Illegal Number
FX	Fix
RW	Reserved Word

C ASCII CODE

DEC	HEX	CODE	DEC	HEX	CODE
32	20		45	2D	-
33	21	!	46	2E	.
34	22	"	47	2F	/
35	23	#	48	30	0
36	24	$	49	31	1
37	25	%	50	32	2
38	26	&	51	33	3
39	27	'	52	34	4
40	28	(	53	35	5
41	29	)	54	36	6
42	2A	*	55	37	7
43	2B	+	56	38	8
44	2C	,	57	39	9

DEC = DECIMAL
HEX = HEXDECIMAL
CODE = CHARACTER

DEC	HEX	CODE	DEC	HEX	CODE
58	3A	:	72	48	H
59	3B	;	73	49	I
60	3C	<	74	4A	J
61	3D	=	75	4B	K
62	3E	>	76	4C	L
63	3F	?	77	4D	M
64	40	@	78	4E	N
65	41	A	79	4F	O
66	42	B	80	50	P
67	43	C	81	51	Q
68	44	D	82	52	R
69	45	E	83	53	S
70	46	F	84	54	T
71	47	G	85	55	U

DEC = DECIMAL
HEX = HEXDECIMAL
CODE = CHARACTER

DEC	HEX	CODE	DEC	HEX	CODE
86	56	V	100	64	d
87	57	W	101	65	e
88	58	X	102	66	f
89	59	Y	103	67	g
90	5A	Z	104	68	h
91	5B	[	105	69	i
92	5C	\	106	6A	j
93	5D	]	107	6B	k
94	5E	^	108	6C	l
95	5F	_	109	6D	m
96	60	`	110	6E	n
97	61	a	111	6F	o
98	62	b	112	70	p
99	63	c	113	71	q

DEC = DECIMAL
HEX = HEXDECIMAL
CODE = CHARACTER

DEC	HEX	CODE	DEC	HEX	CODE
114	72		128	80	
115	73		129	81	
116	74		130	82	
117	75		131	83	
118	76		132	84	
119	77		133	85	
120	78		134	86	
121	79		135	87	
122	7A		136	88	
123	7B		137	89	
124	7C		138	8A	
125	7D		139	8B	
126	7E		140	8C	
127	7F		141	8D	

DEC = DECIMAL
HEX = HEXDECIMAL
CODE = CHARACTER

DEC	HEX	CODE	DEC	HEX	CODE
142	8E		156	9C	
143	8F		157	9D	
144	90		158	9E	
145	91		159	9F	
146	92		160	A0	
147	93		161	A1	
148	94		162	A2	
149	95		163	A3	
150	96		164	A4	
151	97		165	A5	
152	98		166	A6	
153	99		167	A7	
154	9A		168	A8	
155	9B		169	A9	

DEC = DECIMAL
HEX = HEXDECIMAL
CODE = CHARACTER

DEC	HEX	CODE	DEC	HEX	CODE
170	AA		184	B8	
171	AB		185	B9	
172	AC		186	BA	
173	AD		187	BB	
174	AE		188	BC	
175	AF		189	BD	
176	B0		190	BE	
177	B1		191	BF	
178	B2		192	C0	
179	B3		193	C1	
180	B4		194	C2	
181	B5		195	C3	
182	B6		196	C4	
183	B7		197	C5	

DEC = DECIMAL
HEX = HEXDECIMAL
CODE = CHARACTER

DEC	HEX	CODE	DEC	HEX	CODE
198	C6		212	D4	
199	C7		213	D5	
200	C8				
201	C9				
202	CA				
203	CB				
204	CC				
205	CD				
206	CE				
207	CF				
208	D0				
209	D1				
210	D2				
211	D3				

DEC = DECIMAL
HEX = HEXDECIMAL
CODE = CHARACTER